Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Adaptive and Unstructured Mesh Cleaving

Jonathan R. Bronson[a], Shankar P. Sastry[a], Joshua A. Levine[b], Ross T. Whitaker[a]

[a]*Scientific Computing and Imaging Institute, Salt Lake City, UT, U.S.A.*
[b]*School of Computing, Clemson University, Clemson, SC, U.S.A.*

Abstract

We propose a new strategy for boundary conforming meshing that decouples the problem of building tetrahedra of proper size and shape from the problem of conforming to complex, non-manifold boundaries. This approach is motivated by the observation that while several methods exist for adaptive tetrahedral meshing, they typically have difficulty at geometric boundaries. The proposed strategy avoids this conflict by extracting the boundary conforming constraint into a secondary step. We first build a background mesh having a desired set of tetrahedral properties, and then use a generalized stenciling method to divide, or "cleave", these elements to get a set of conforming tetrahedra, while limiting the impacts cleaving has on element quality. In developing this new framework, we make several technical contributions including a new method for building graded tetrahedral meshes as well as a generalization of the isosurface stuffing and lattice cleaving algorithms to unstructured background meshes.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: Meshing ; Tetrahedral ; Multimaterial ; Unstructured ; Bounded ; Quality

1. Introduction

The automatic construction of volumetric meshes has for decades been a relevant problem for fields spanning computational science and engineering to computer graphics. Meshes enable finite-element simulations, visualization, and in general the digitization of physical phenomena. Despite a great deal of important research and many fundamental advances, the general problem of tetrahedral meshing remains unsolved and challenging. Three of the most desirable properties for meshes are to have adaptive element size, good tetrahedral quality, and an adequate representation of boundaries. Taken together, the constraints imposed by these properties are often in conflict with each other. For example, meshing with perfectly isotropic elements strictly prevents the ability to vary element size, since gradedness would require mesh edges of differing lengths. Other constraints may act in concert, for example meshes that adapt to the curvature tensor of a smooth surface may naturally induce elements that better approximate the surface.

In this work, we focus on the problem of building tetrahedral meshes that conform to a geometric boundary without sacrificing other desirable characteristics of a mesh. We say a mesh *conforms to a boundary* if both (1) a collection of mesh vertices lies on the boundary and (2) a collection of mesh simplices passing through these vertices sufficiently approximates the boundary. Our specific focus is on boundaries that are piecewise smooth manifolds: collections of smooth surface patches that meet in potentially non-manifold configurations along a network of curve segments. Such shapes are represented as the surfaces which lie on the boundary of different volumetric materials, such as those encountered in segmented 3D images or between different phases in multiphase flow.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

We take the approach of explicitly decoupling the problem of conforming the mesh and delay its resolution. Instead, we initially construct a volumetric *background* mesh that satisfies all other desirable properties. This approach is a small but novel change from most boundary conforming meshing algorithms. By restricting the problem, we design a modular meshing pipeline, where mesh adaptation can be completed independently from the task of boundary representation. We experiment with an electrostatic particle formulation to generate appropriate background meshes by adapting vertex locations to a sizing field defined based on a distance transform to material boundaries.

Note that while this background mesh formulation leads to good gradedness and reasonable angles, its use is a free choice in our pipeline. We make no specific assumptions about the background mesh for later stages, and as such our pipeline has enough flexibility for mesh adaptations driven by other criteria. Given the background mesh, we next apply a single *cleaving* step that conforms the mesh to a boundary. This cleaving step is a novel generalization of both the isosurface stuffing [22] and lattice cleaving algorithms [8]. Unlike past work, our approach relaxes the requirement that the background meshes are body-centered cubic (BCC) lattices.

The initial contribution of mesh cleaving schemes focused on building meshes with the highest quality dihedral angle bounds. An alternate perspective is that a mesh cleaving step should judiciously limit the change in prescribed qualities of an input mesh. Near where we cleave, we do expect (and experience) some quality degradation, so if the background mesh has small angles already they may worsen. Nevertheless, the approach effectively separates concerns and by deferring the boundary meshing step problem of satisfying multiplied constraints is simplified. In this work we show that for both structured and unstructured background meshes, we can still cleave while limiting changes in element quality.

1.1. Related Work

Tetrahedral meshing in the presence of boundaries is a well-studied problem in the literature, we review only the most relevant papers to our approach. For a more complete overview of field, we recommend a recent survey by Shewchuk [32].

With the exception of lattice-based approaches [22,30,38], when meshing to conform to a boundary the majority of algorithms first try to capture the boundary constraint. Delaunay refinement [12] is one such example. Typically, such meshes are produced by inserting vertices in boundary features in an increasing dimensionality and thus increasing complexity. Alternatively, some meshing techniques assume that an input boundary mesh is given, and a conforming mesh is built (through insertions and flips) such that every boundary element exists in the output or is a union of output elements [17,19,36]. Working with an input boundary mesh is also natural for advancing front techniques [25,27], since the boundary elements provide a seed surface from which to grow the front. We remark that an interesting conclusion, parallel to our own, in the domain of hexahedral meshing by advancing front (paving and plastering) is that relaxing the boundary constraint by delaying boundary meshing leads to overall improved results [34].

Meshing by sequencing through boundaries of increasing dimension is particularly popular for tetrahedral meshing of the multimaterial domains we consider. The representation of complex non-manifold boundaries is the major challenge, so it is popular to tackle it first (e.g. in both Delaunay and variational methods, or combinations of them [5,7,9,28,35]). However, as dimension increases, the fixed collection of lower dimensional elements impose an increasingly complex set of constraints for the next stage of meshing. In terms of Delaunay refinement, this typically means that the placement of locations for new vertices becomes limited. The insertion of points to improve elements may be blocked since they would otherwise disturb boundary features [5,9,10]. In terms of variational approaches, this means that the next stage of optimization becomes a more expensive constrained energy minimization [1,7]. Alternatively, one can allow the boundary to be disturbed, but then iterate in attempt to reconform to the boundary [35]. Nevertheless, the Delaunay-based techniques often offer provable guarantees on the ability to conform to piecewise linear [31], smooth [4,11], and piecewise-smooth [5,9] boundaries.

By comparison, approaches that start with a background lattice are presented with the opposite challenge. In the absence of a boundary constraint, it may be possible to mesh volumetrically while satisfying a broad range of constraints. However, proving one can still capture the boundary becomes more complex. Typically, these approaches use a highly structured lattice to rapidly construct meshes with are self-similar, such as an octree [30,38]. While what has been proved about lattice-based algorithms is limited, there is a growing body of work. For 2D domains, it was shown that a quadtree can be provably adapted to lead to a capturing a polyhedral domain [3]. More recent approaches apply the BCC lattice, which is not only a naturally good domain for approximating trivariate functions [21], but also

a Delaunay triangulation with good dihedral angles everywhere. Labelle and Shewchuk were the first to use BCC lattices as background meshes to build tetrahedral meshes that conform to a smooth boundary while maintaining dihedral angle bounds [22]. Doran et al. experiment with using acute lattices instead of BCC [15]. Algorithms such as those of Zhang et al. [39], Chernikov and Chrisochoides [13], and Liu et al. [24] extend some of these ideas to the case of multimaterial medical domains with significant experimental results. Most recently, Liang and Zhang prove bounds on constructing adaptive meshes which conform to smooth surfaces [23]. Bronson et al. [8] generalize the results of Labelle and Shewchuk with their lattice cleaving approach, and were also able to generalize a proof that in the case of multimaterial boundaries, a bound for the dihedral angles of the resulting elements exists. In this work, we generalize the lattice cleaving technique to arbitrary background grids with a broader range of input characteristics.

2. Methodology

The strategy we propose is to separate the creation and adaptivity of quality volume elements from surface conforming constraints. This separation can be achieved through a volumetric meshing pipeline (Fig. 1). First, the desired and necessary element characteristics throughout the volume must be determined. These constraints are then used as input to a meshing algorithm to generate an ambient *background* mesh. This mesh will know nothing of material interfaces, but will have appropriately sized elements thanks to the sizing field driving it. Finally, this background mesh will be fed to the cleaving algorithm, where elements near material interfaces will be cleaved to conform to these surfaces. We provide algorithms which are separately capable of handling the specific pieces of this pipeline.

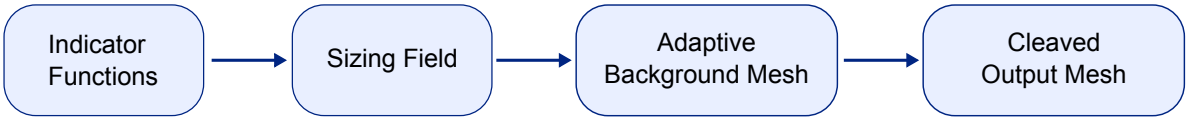

Fig. 1. Proposed meshing pipeline for conforming volumetric meshing.

2.1. Local Feature Size and Sizing Fields

Mesh elements must be adaptively sized for the purpose of geometric fidelity and PDE solution accuracy while simultaneously reducing the number of elements needed for an accurate numerical simulation. A *sizing field* is a scalar field that at every point dictates the ideal size of an element centered around that point. We suggest that a suitable sizing field should posses the following properties: a) it should be small near thin features and high-curvature regions; b) it should progressively increase for larger features and lower-curvature regions; c) it should be sufficiently large at points that are far from material interfaces; and d) it should satisfy Lipschitz continuity conditions. Abrupt changes in the sizing field is undesirable because the quality of the resulting elements is likely to be poor.

In surface mesh generation algorithms, the concept of *feature size* has been widely used to accurately capture the topology of the object that is being meshed. It is defined only on the surface of the object, and it is defined as the distance from the *medial surface* of the object. The medial surface is a surface formed by those points that have more than one closest points on the object boundary. It is also referred to as the "skeletonization" an object. Thin features have a small feature size, and large features have a large feature size.

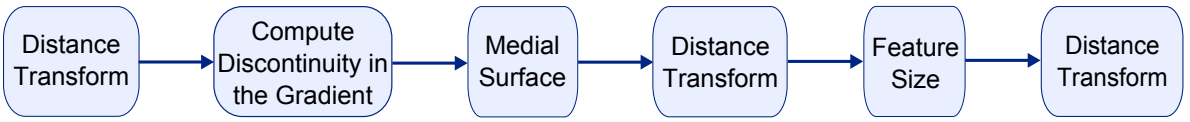

Fig. 2. Proposed pipeline for sizing field computation.

The feature size defined on the surface of an object can be extended over the whole domain to dictate the size of elements in every region of the domain. Persson [29] describes an algorithm that uses a variant of *distance transform* computed from the surface of the object using the feature size as the initial set of values for the distance transform.

The distance transform is a scalar field that specifies the distance to the closest point, curve, or surface. It can be visualized as a series of wave fronts emanating from the given set of points, curves, or surfaces. Figure 3 provides an illustration of the distance transform of a C-shaped object and its medial surface (medial axis, in 2D). Notice that the discontinuity in the gradient of the distance transform indicates the medial surface.

Our algorithm to compute a sizing field is based on the work of [6] and [29], and it is performed on a voxel domain with sub-voxel accuracy. A pipeline describing our algorithm is shown in Fig. 2. The technique relies on solving for three distance transforms over this domain. First, the distance transform is computed starting from the material boundary surfaces. Because the distance transform is nonsmooth only where the wave fronts collide and is linear otherwise, we can use the Hessian to compute the set of points on the medial surface. The Hessian vanishes at all locations except at the medial surface. The voxels where the Hessian does not vanish define the medial surface. Next, the distance transform is computed again from the medial surface. The values of the distance transform at the boundary locations thus define the feature size at those points. Finally, the distance transform is computed again from the boundary. This time, the initial value at the boundary is set as the feature size computed in the previous step and the gradient of the distance transform is limited as a user parameter. This gives us the sizing field over the whole domain.

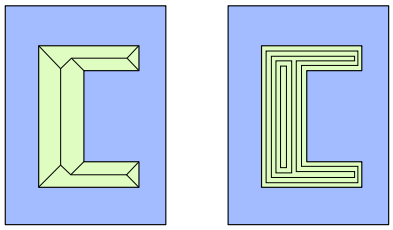

Fig. 3. Left: The medial surface (axis) of a C-shaped object. Right: The distance transform level sets. Note, the distance transform and the medial axis are also computed outside the object.

To initialize the starting value for the first distance transform, we have to find the approximate locations of material interfaces on the edges of the voxel grid. Our distance transform solver uses the fast marching method (FMM) [26] that is second-order accurate is used with a heap-based priority queue. Note that a $3 \times 3 \times 3$ stencil is required to compute the Hessian. Thus, the resolution of the grid should be appropriately chosen to capture the topology and features of the required size. As we shall describe later, we construct both structured and unstructured meshes using the sizing field. We guarantee that the topology is correctly captured only for the structured mesh. Since the unstructured meshes is constructed using a heuristic algorithm, the topology may not be correctly captured by the mesh.

2.2. Electrostatic Particle Distributions

Particle systems are often used in mesh generation algorithms in order to obtain a distribution of points satisfying certain constraints. For instance, the idea of *centroidal Voronoi diagrams* has been used in several mesh generation algorithms (eg. [16]) for isotropic point distribution requirements. The concept has been extended for anisotropic metrics [18] as well. Other notable examples of the use of particle system in mesh generation include Hart's *et al.'s* surface sampling technique [20], Yamakawa and Shimada's ellipsoidal bubble packing algorithm[37] and Meyer *et al.'s* particle sampling technique for multimaterial meshing [28].

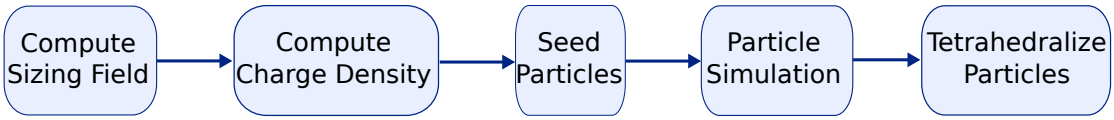

Fig. 4. Proposed pipeline for generating an adaptive, unstructured background mesh.

In each of these techniques, a stable point distribution is achieved by iteratively minimizing an energy function that matches the meshing requirements. These approaches typically distribute particles on material interfaces first, using these surfaces as a constraint for a volumetric meshing algorithm. In contrast, our new approach to adaptive mesh generation distributes particles directly over the whole of the domain, without any regard for material interfaces. These points are provided to a delaunay tetrahedralization algorithm to build a background mesh suitable for the next stage of our new meshing pipeline.

We propose a new electrostatic particle simulation technique to distribute particles over the whole input domain. A pipeline describing our algorithm for generating an unstructured background mesh is shown in Fig. 4. Unlike typical electrostatic particle systems, in which particles tend to gather at protrusions and sharp feature, this new

approach produces particle distributions that accurately match the input sizing field specification. Particles are given an electrostatic charge as usual (see [28] and [37]), but the domain itself is also given an opposite charge.

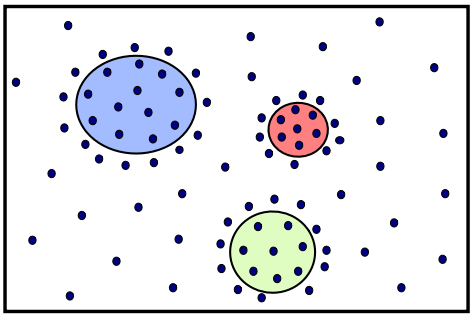

Fig. 5. An illustration of a particle distribution over a domain.

This background charge interacts with particles in such a way that a net-zero charge can only be produced when the set of particles precisely matches the desired input sizing field. Our particle distribution technique is designed such that the regions of the domain with a small feature size will have a large charge density, while regions with larger feature size will have a smaller charge density. This ensures that greater number of particles congregate in the region where they are desired. Figure 5 provides a pictorial illustration of a particle system where more particles are needed in the three circular regions.

The force on a particle in the domain is a sum of both the background charge density, as well as the other particles in the domain. The force due the background charge density is determined by computing the gradient of the electrostatic potential due to the charge density, and the force due other particles is computed by summing up the forces from all other particles in the domain. The particles are moved based on the particle-field and particle-particle interactions. When the system converges to a stable equilibrium, the distribution will respect the provided sizing field.

2.2.1. Charge Density Computation

In order to compute the charge density corresponding to an input sizing field, we exploit the relationship between a sizing field and optimal sphere packing density. The optimal sphere packing density (fraction of volume filled by spheres) for a hexagonal close packing is $\eta = \frac{\pi}{3\sqrt{2}}$. Let the volume of the domain be V, and assume a uniform sizing field, l, in the domain. The number of spheres, n, of the radius $l/2$ in the volume is given by

$$n = \frac{\eta V}{\frac{4}{3}\pi\left(\frac{l}{2}\right)^3}. \tag{1}$$

We set the number of particles to be the nearest integer to the total background charge in the domain (computed by integrating the charge density). In this way, we realize a stable equilibrium where negatively-charged particles neutralize the positively-charged background. Therefore, we want $n = \rho V$, where ρ is the charge density. Solving for ρ, we obtain $\rho = \sqrt{2}/l^3$. Thus, the charge density is set to be inversely proportional to the cube of the sizing field. Note that any monotonically decreasing charge density with respect to the sizing field is likely to yield some results that respect the adaptivity requirements, but its feasibility depends upon the application. The charge density ρ is usually not a constant over the whole domain. Therefore, we generalize the number of particles over the whole domain to be $n = \int_\Omega \rho dV$, where Ω is the domain.

2.2.2. Potential and its Gradient Computation

The electrostatic potential due to the background charge density is computed by solving the Poisson equation, $\nabla^2 u(x) = f(x)$, where u is the electrostatic potential and f is the charge density. The Poisson equations are solved using the finite difference scheme on a structured grid with Dirichlet boundary conditions computed by summing up the potential due to the charge density at every point in the boundary. This process is accelerated using an octree-based technique. We use the linear conjugate gradient solver to determine the solution of the linear system resulting from the finite-difference approximation of the equation and boundary conditions.

In order to compute the local charge density and its gradient at any point in the domain, we utilize a cubic convolution-based interpolation technique over the structured grid. Its main advantage over a trilinear interpolation approach is the continuity of the gradient of the interpolated function over the whole domain, which helps the system avoid local minima. Additionally, the analytical gradient of the interpolant can be accurately computed. When trilinear interpolation is used instead, the particles get "trapped" in the faces, edges, and corners of the cubic elements of the grid. This naturally results in a suboptimal point distribution.

2.2.3. Electrostatic Simulation

For the electrostatic simulation, the number of particles are seeded in the various part of the domain is proportional to the local charge density. This ensures a quick convergence of the particle system since the movement of particles is locally restricted. Each individual particle is separately moved by a distance proportional to the force on the particle and the step size for that iteration. We adaptively vary the step size in each iteration. The process is accelerated using the octree-based Barnes-Hut simulation technique [2].

Each octant of the octree stores the location of the center of the charge distribution within the cell, as well as the total contained charge. The force on a particle charge at any location is computed by traversing down the octree. If the ratio of the length of the smallest side of an octant vs the distance between the particle charge and the center of the charge distribution is lower than a threshold θ, the force due to all the charges in the octant on the particle charge is approximated by a single point. This significantly reduces the number of floating operations required to compute the force on the particle charge. The particles are iteratively moved to optimal locations based on the forces acting upon them. Once a static equilibrium is reached, the particles are tetrahedralized using Tetgen [33].

2.3. Unstructured Cleaving

In order to produce a surface conforming mesh from an unstructured background mesh, we turn to a technique which, up until now, has been demonstrated to work only on regular lattices. Lattice Cleaving, like Isosurface Stuffing, is a stencil-based technique for producing conforming tetrahedral meshes with elements of bounded quality.

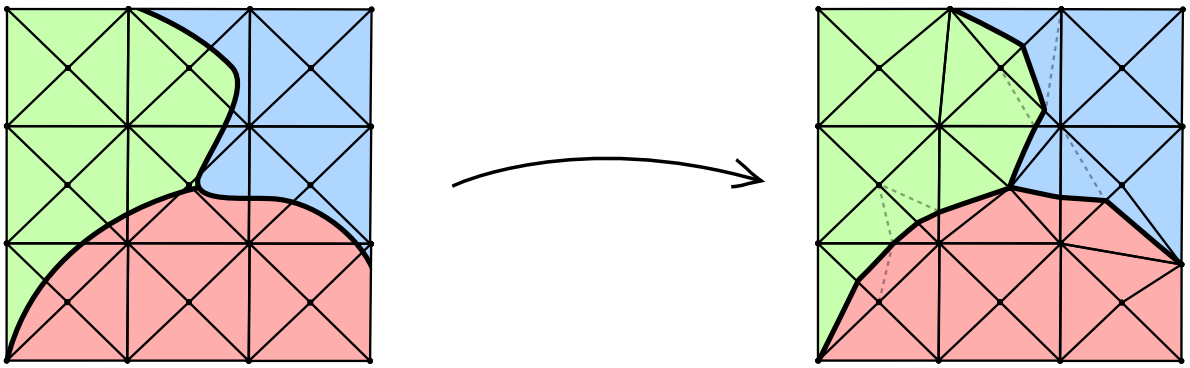

Fig. 6. Illustration of lattice cleaving in 2D, Left: background mesh with material interfaces overlaid, Right: cleaved output

In one sense, these algorithms can be considered mesh *processing* algorithms. They take as input a mesh, a regular lattice of high quality tetrahedra (the Body-Centered Cubic (BCC) Lattice), and through a series of vertex warps and stencil operations transform the mesh elements to conform to material boundaries (Fig 6). We use this paradigm to extend the technique to arbitrary unstructured and irregular background *input* meshes. We do this in the context of multimaterial volumes. This section details the technical considerations that need to be accounted for to achieve this generalization.

The basis of the lattice cleaving algorithm remains unchanged. The input elements are still tetrahedra, the violation snapping and warping rules carry over, and the same output stencil set is used. However, there are two fundamental challenges when moving to an unstructured mesh: resolving stencil consistency across the shared face of neighboring background tetrahedra, and alpha parameter selection.

2.3.1. Stencil Consistency and Generalization

Some output topologies have multiple permissible tessellations. Without consideration for consistency, two neighboring tetrahedra may stencil their shared face differently, leading to a topological hole in the mesh. This problem can be avoided by carefully orienting each background element before applying output stencils. Isosurface stuffing does this using a simple parity rule that exploits the regular structure of the lattice. As we're interested in unstructured meshes, we cannot rely on any predetermined structure of an input mesh. Instead, we take inspiration from the approach used in the lattice cleaving algorithm.

Rather than rely on the background lattice structure, the lattice cleaving algorithm resolves stencil consistency by taking advantage of the fact that all output stencils can be generated through a series of material collapses of the most complex 4-material stencil. This mapping is called a *generalization*. In this model, a set of stencil outputs is consistent if there exists a set of *virtual* interfaces that when snapped and warped, would have led to that stencil set. The lattice cleaving algorithm for placing these virtual interfaces and their snap destinations is specific to the BCC lattice and so we develop a more versatile algorithm.

We observe that if you take a three-material face, the only way a set of cuts cannot be snapped to collapse into a simpler stencil form is when their movement forms a cycle (Fig. 7). We further observe that for any set of valid face generalizations on a tetrahedron, there is always a way to move a virtual quadruple point to obtain a valid stencil.

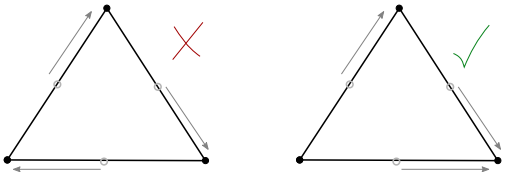

Fig. 7. Left: Cyclic virtual cuts lead to an unsatisfiable generalization. Right: Any ordered priority can lead to safe generalization.

One way to guarantee a set of virtual cuts never move in a cycle is to enforce an ordering. If each vertex is given an integer id, then enforcing the rule that a virtual cut always moves to the higher (or lower) vertex, is sufficient. Most mesh implementations store an ordered list of vertices anyway, so creating this order is trivial. The remaining work is determining the destination of virtual triple points and virtual quadruple points. Since the triple point of a face is shared, any valid destination will by definition be consistent across the face, and since quadruple points exist on the interior of tetrahedron, any valid destination will suffice.

There are two principles to selecting valid virtual interface locations: Virtual interfaces always snap to the next smallest simplex with the most colocated virtual interfaces; and ties are always settled in favor of real interfaces over virtual interfaces. Figure 8 illustrates the two ways this manifests on a background lattice face. If one virtual cut exists, the virtual triple snaps to the real cut on the edge incident to the snapped virtual cut. This collapses the missing third material region onto the edge. If two virtual cuts exist, the virtual triple snaps to the colocation of the two virtual cuts (i.e., the vertex with the smallest id). Similarly, the missing materials collapse onto an edge and onto a vertex.

Fig. 8. This figure illustrates the generalization of one and two material face stencils. (left) A two material stencil is generalized. The virtual cut on the edge connecting vertex 0 and vertex 1 moves to vertex 0. The virtual triple follows the cut onto the edge connecting vertex 0 and vertex 2. (right) A one material stencil is generalized. All virtual cuts move to the adjacent vertices with the lowest index. The virtual triple follows the virtual cuts that end up on the same vertex.

2.3.2. Alpha Selection

In the lattice cleaving and isosurface stuffing algorithms, the alpha parameter controls the trade off between stenciling and warping. It does this by defining the regions in which an interface is considered to be *violating*, and will need to be snapped. Labelle and Shewchuk utilized an automated computational proof to determine optimal alpha parameters for the long and short edges of the BCC lattice in isosurface stuffing. For the multimaterial case, the state space for this proof becomes computational infeasible, and so the authors provide a theoretical proof of bounds, and utilize conservative version of the parameters from the two material case.

As we move to an unstructured mesh, the challenge of finding optimal alpha parameters becomes even more difficult. While the BCC lattice has two edge lengths and symmetric elements, any given unstructured mesh may have no two neighbors with identical edge lengths or element shapes. So rather than having to choose two alpha values, short and long, the user must pick up to e alpha values. The set of optimal alpha values is therefore unique for every input mesh, and must be computed at run-time.

We present an algorithm for computing conservative alpha violation parameters that allow the quality proof of lattice cleaving to still hold. This algorithm has the benefit of parameterizing the alpha values of each edge by a single global alpha value.

First, we observe that in the limit, as α approaches zero, the background mesh stops all warps and stencil elements can become arbitrarily bad. As α increases, the snapping and warping procedure becomes increasingly aggressive, ultimately to the point where it is unsafe and may result in degenerate elements.

If we permit the four vertices of a tetrahedron to move in any direction, the shortest distance they can travel before the element becomes degenerate (coplanar) is along the shortest vertex altitude. If we show no preference to any particular vertex, they meet in the plane that is halfway along the altitude, or $\frac{h}{2}$ from each vertex, where h is the height of the altitude. Figure 9 illustrates this in 2D. This observation provides an upper limit for how aggressive our α selection can be for any particular vertex with respect to a tetrahedron to which it belongs. We can then parameterize over this space as $\alpha = (\frac{1}{2} - \xi)h$ for $0 < \xi < \frac{1}{2}$ and provide ξ as a user parameter to optimize.

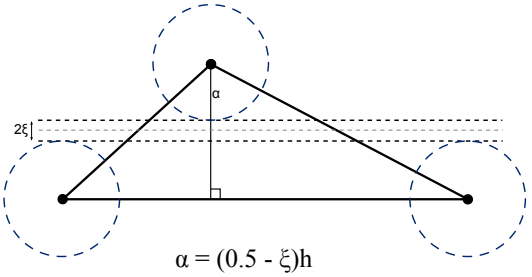

Fig. 9. Illustration of how the maximum safe distance that a vertex may move is bounded by the height of the corresponding altitude.

The algorithm for computing a set of α parameters is thus as follows: Begin with current best guess for ξ. For each vertex v_i, iterate over all incident tetrahedra and set $\alpha_i = \min\left\{\alpha_i, (\frac{1}{2} - \xi)h\right\}$.

In this formulation, the α parameters around a vertex are all the same and are stored on the vertex, rather than on the edge. This algorithm can be used in conjunction with the proof of bounds from the Lattice Cleaving algorithm [8] to prove that it also places bounds on the quality of output elements. We replace Lemma 3 of the proof with the following alternate lemma.

Lemma 3. *For every tetrahedron with ϵ-good dihedral angles, there exists a space of permissible violation parameters $\bar{\alpha}$ such that the tetrahedron will retain ϵ-good angles after warping.*

Proof. Let t be a background tetrahedron with ϵ-good angles. Whether measured by aspect ratio or dihedral angle, the tetrahedron decreases in quality towards degeneracy as the vertices approach becoming coplanar. The shortest path vertices can move during a warp to create such a coplanarity is along the shortest altitude. Therefore, any safe set of α values must follow the inequality

$$\alpha_1 + \max\{\alpha_2, \alpha_3, \alpha_4\} < h, \tag{2}$$

where α_i is an α-ball around vertex v_i and v_1 is the vertex with the smallest altitude. This inequality is easy to satisfy and can be parameterized as

$$\alpha = (\frac{1}{2} - \xi)h \text{ for } 0 < \xi < \frac{1}{2}. \tag{3}$$

As ξ approaches $\frac{1}{2}$ the maximum safe *alpha* values are reached. As ξ approaches 0 warping becomes increasingly restricted.

□

3. Results and Discussion

Together, the contributions of this paper offer a full multimaterial volumetric meshing pipeline. In this section, we illustrate what such a system is capable of through the use of both synthetic and real-world data. As discussed in Section 2.2, our electrostatic particle formulation, while extendable to further optimization, currently only optimizes vertex positions and may produce low volume elements. For comparison, for each example dataset, we generate results using both the electrostatic background mesh, as well as a structured, adaptive octree mesh. In this way, the

cleaving of these meshes may more easily be seen as a mesh processing procedure, with results for both input meshes compared side by side.

Our implementation for creating the adaptive octree background meshes is straight forward. The octree begins with a single cell that encloses the whole domain. The algorithm queries a sizing field oracle that returns the minimum sizing field within the cell. If this size is smaller than the width of the cell, the tree subdivides. This routine is ran recursively until the smallest local feature size is no smaller than half a cell. Then, the graded stencil set from [8] and [22] is used to fill in the tree and output the background mesh.

We ran our experiments on a single core of the 16-core AMD Opteron 8360 SE 2.5GHz processor with 96GB of RAM running the openSUSE 11.3 (x86_64) operating system with gcc (SUSE Linux) 4.5.0 20100604 compiler. The size of the mesh background and the output generated by our implementation is reported in Table 1. Table 2 reports the Hausdorff distance between the surfaces meshes when structured and unstructured meshes are generated using the respective algorithms. They were computed in Meshlab [14] by sampling points on one of the surface meshes and computing the distance to the corresponding point on the other surface mesh. The timing for each stage of the pipeline for structured and unstructured meshes are reported in Table 3 and 4, respectively. In our particle simulation, the threshold θ (see Section 2.2) was reduced from 1 to 0.25 in 200 iterations and held at 0.25 for the next 100 iterations. The time taken for the first 200 iteration is roughly one-third the time reported in Table 4.

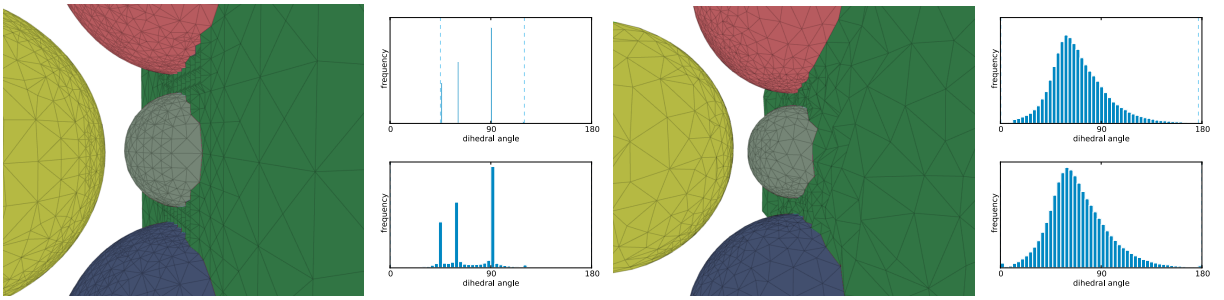

Fig. 10. A set of four sphere materials embedded in a green background material. (left) structured background mesh (right) unstructured background mesh

Figure 10 contains a synthetic dataset of various spheres. These spheres can each be independently meshed with a fewer number of tetrahedra, but together produce small cavities that drive the sizing field down. The surfaces and background meshes adapt to this sizing field as necessary. The left result is generated from a graded octree background mesh, and the right result is generated from a graded electrostatic mesh.

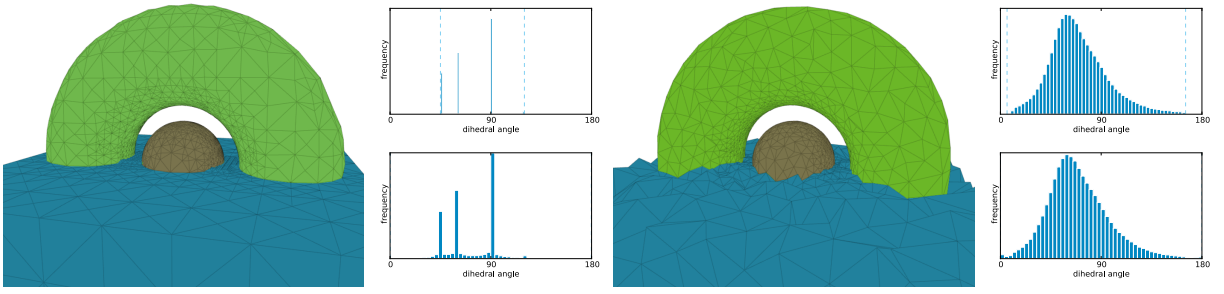

Fig. 11. A torus with a sphere inside it, embedded in a blue background material. (left) structured background mesh (right) unstructured background mesh

Figure 11 contains two meshes generated from synthetic dataset of a torus and a sphere in close proximity. Grading along the surfaces as well as the background mesh can be seen in both the structured and unstructured meshes. Figure 12 contains two meshes generated from an MRI of a human torso. The clear cut of the boundary of the domain can be seen on both the octree and electrostatic background meshes. Because the sizing field goes to zero at three-material junctions, user settings limit the sizing field in these regions. The time taken for each stage of pipeline is reasonable for the size of the domain and the size of the mesh needed to capture all the features in the domain.

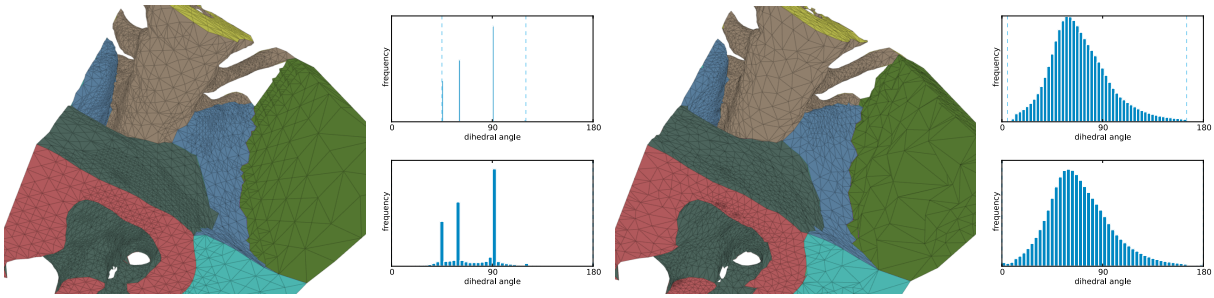

Fig. 12. Section of human torso MRI, embedded in a green background material, with (left) structured and (right) unstructured backgrounds meshes.

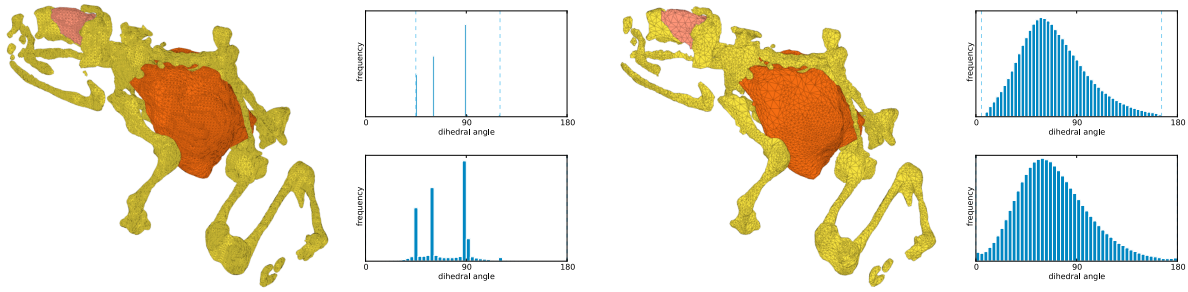

Fig. 13. An MRI of a frog with (left) structured and (right) unstructured background meshes.

Figure 13 contains two meshes generated from an MRI scan of a frog. The structured version of this mesh ends up being much larger due to the octree dimensions enforcing power-of-two dimensions. The unstructured background mesh has no such requirements. At the same time, the user chosen scaling factor is so large that some fine features near the mouth have been missed in the unstructured mesh. A less aggressive scaling would not cause this kind of loss. Also note that the time taken to compute the sizing field is much larger for this domain than for other domains. This is because the size of the domain and the corresponding large grid on which the distance transforms are computed. The time taken to solve the Poisson equation is also large for the same reason. This can be accelerated using a suitable preconditioner such as the algebraic multigrid preconditioner. Although this domain needs more vertices to capture all the features present in the data than the torso dataset, the time taken to execute the accelerated particle system is much lower. This is due to the large size of the domain. As the particles are distributed further away from each other, the Barnes-Hut algorithm is able to approximate the forces from groups of particles that are at large distances from a particle in consideration.

Since our particle distribution technique is a global technique, i.e., the position of a particle depends on the position of all other particles (not just its neighbors), it is less likely to get stuck in a local optima. In each of our test cases, we were able to control of the number of vertices in a given region of a domain is a precise manner. While other refinement or variational techniques incrementally add vertices, this technique can determine the number of particles *a priori*. As it is slow, in real applications, we recommend running only a few iterations of the particle distribution scheme and using local techniques to optimize the positions of the particle or to improve the quality of the resulting background mesh.

Examining the angle distributions across each example dataset, we see that the cleaving operation largely preserves the angle profile of each input, with the distributions spreading outward and toward the tail ends. The structured meshes begin with much higher quality elements and are therefore more resilient to the effects of cleaving. For this reason, it's especially important to create high quality background meshes. Improved alpha value selection would also help to mitigate the effects of cleaving.

In this work, we have illustrated the potential power of decoupling the problems of mesh element and boundary constraints. The particle system presented achieves its goals with simplicity due to the lack of interface surfaces, and is simply one method for generating unstructured background meshes. One possible alternative algorithm to use for

Domain	Size	Mesh Type	Background Mesh		Output Mesh	
			#Vertices	# Elements	#Vertices	# Elements
Spheres	[64, 64, 64]	S	26,219	129,804	26,921	133,858
		U	11,856	71,160	12,500	74,050
Torus	[64, 64, 64]	S	35,082	175,456	35,951	180,238
		U	13,556	82,298	14,259	85,626
Torso	[64, 64, 64]	S	145,240	721,678	149,818	746,955
		U	60,867	360,182	64,826	378,361
Frog	[260, 245, 150]	S	1,057,586	5,347,544	1,087,272	5,515,823
		U	70,415	428,173	74,489	447,741

Table 1. The size of the domain and the sizes of the background and output meshes. S denotes structured meshes, and U denotes unstructured meshes.

Domain	# Sampling Points	Hausdorff Distance			
		min.	mean	r.m.s.	max.
Spheres	75,977	0.0	0.051	0.101	1.927
Torus	56,052	0.0	0.076	0.144	1.063
Torso	312,812	0.0	0.056	0.111	2.589
Frog	2,886,382	0.0	0.214	0.451	9.650

Table 2. The Hausdorff distance between the structured and unstructured meshes for each of the data set. They have been computed by sampling points on one of the surface meshes and finding the distance to the corresponding point on the other surface mesh in Meshlab [14].

Domain	Time (in seconds)		
	Sizing Field	Background Mesh Creation	Cleaving
Spheres	2.39	7	10
Torus	2.15	6	15
Torso	4.50	43	97
Frog	288.9	520	202

Table 3. The time taken for each stage of the pipeline to generate the structured meshes.

Domain	Time (in seconds)			
	Poisson Solver	Particle System	Tetgen	Cleaving
Spheres	21	2,103	1.30	5
Torus	22	2,466	1.40	5
Torso	22	17,727	2.24	45
Frog	1,742	9,474	2.75	50

Table 4. The time taken for each stage of the pipeline to generate the unstructured meshes. Note the time taken to generate the sizing field is not included in the table because it has already been included in Table 2 for structured meshes.

the background mesh generation of the pipeline is centroidal Voronoi tessellation (CVT). This technique is interesting because with variable metrics (as presented in [16]) it has the ability to push low volume elements to the boundary of the domain. This is ideal for the cleaving algorithm, since these external elements will ultimately be discarded.

We have also demonstrated that the lattice cleaving algorithm is extendable to unstructured meshes in a straightforward manner. The method we chose for producing safe α parameters, though conservative, provides a starting point for more sophisticated methods. Its major drawback is being symmetric around a vertex. This means that as the difference in relative size of neighbor tetrahedra increases, the parameters will be increasingly more conservative. Detaching this symmetry constraint would be a significant step towards solving for truly optimal α parameters. As a whole, this work suggests that the union of traditional and combinatoric meshing techniques promises to provide a fertile ground for new developments in high quality conforming mesh generation for unstructured meshes.

Acknowledgments

We would like to thank the US National Science Foundation for its support under award IIS-1314757, the National Institute of General Medical Sciences of the National Institutes of Health (NIH/NIGMS) for its support under grant number P41GM103545, the NIH/NIGMS Center for Integrative Biomedical Computing for its support under grant number 2P41 RR0112553-12, and the Department of Energy for its support under grant number NET DE-EE0004449. We would also like to thank Mark Kim for providing us with a base for the Barnes-Hut acceleration structure.

References

[1] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph.*, 24(3):617–625, 2005.
[2] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324:446–449, 1986.
[3] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. In *Proceedings of Foundations of Computer Science*, pages 231–241. IEEE, 1990.
[4] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graph. Models*, 67(5):405–451, 2005.
[5] D. Boltcheva, M. Yvinec, and J.-D. Boissonnat. Feature preserving Delaunay mesh generation from 3D multi-material images. *Comput. Graph. Forum*, 28(5):1455–1464, 2009.
[6] S. Bouix and K. Siddiqi. Divergence-based medial surfaces. In *In Sixth European Conference on Computer Vision*, pages 603–618, 2000.
[7] J. R. Bronson, J. A. Levine, and R. T. Whitaker. Particle systems for adaptive, isotropic meshing of CAD models. In *19th IMR*, pages 279–296, Oct. 2010.
[8] J. R. Bronson, J. A. Levine, and R. T. Whitaker. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *21st IMR*, pages 191–210, Oct. 2012.
[9] S.-W. Cheng, T. K. Dey, and J. A. Levine. A practical Delaunay meshing algorithm for a large class of domains. In *16th IMR*, pages 477–494, 2007.
[10] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. *Discrete Comput. Geom.*, 43(1):121–166, 2010.
[11] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *SIAM J. Comput.*, 37(4):1199–1227, 2007.
[12] S.-W. Cheng, T. K. Dey, and J. R. Shewchuk. *Delaunay Mesh Generation*. Chapman and Hall / CRC computer and information science series. CRC Press, 2013.
[13] A. N. Chernikov and N. Chrisochoides. Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. *SIAM J. Sci. Comput.*, 33(6):3491–3508, 2011.
[14] P. Cignoni, M. Corsini, and G. Ranzuglia. Meshlab: an open-source 3D mesh processing system. *ERCIM News*, (73):45–46, 2008.
[15] C. Doran, A. Chang, and R. Bridson. Isosurface stuffing improved: acute lattices and feature matching. In *SIGGRAPH Talks*, page 38, 2013.
[16] Q. Du and D. Wang. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *Int. J. Num. Meth. Eng.*, 56(9):1355–1373, 2003.
[17] Q. Du and D. Wang. Boundary recovery for three dimensional conforming delaunay triangulation. *Comput. Meth. Appl. Mech. Eng.*, 193(2326):2547 – 2563, 2004.
[18] Q. Du and D. Wang. Anisotropic centroidal Voronoi tessellations and their applications. *SIAM J. Sci. Comput.*, 26(3):737–761, 2005.
[19] P. L. George, H. Borouchaki, and E. Saltel. ultimate robustness in meshing an arbitrary polyhedron. *Int. J. Num. Meth. Eng.*, 58(7):1061–1089, 2003.
[20] J. C. Hart, E. Bachta, W. Jarosz, and T. Fleury. Using particles to sample and control more complex implicit surfaces. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.
[21] Z. Hossain, U. R. Alim, and T. Möller. Toward high-quality gradient estimation on regular lattices. *IEEE Trans. Vis. Comput. Graph.*, 17(4):426–439, 2011.
[22] F. Labelle and J. R. Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph.*, 26(3):57, 2007.
[23] X. Liang and Y. Zhang. An octree-based dual contouring method for triangular and tetrahedral mesh generation with guaranteed angle range. *Eng. Comput.*, 2013. Accepted.
[24] Y. Liu, P. Foteinos, A. Chernikov, and N. Chrisochoides. Mesh deformation-based multi-tissue mesh generation for brain images. *Eng. Comput.*, 28(4):305–318, 2012.
[25] R. Löhner and P. Parikh. Generation of three-dimensional unstructured grids by the advancing-front method. *Int. J. Num. Meth. Fluid*, 8(10), 1988.
[26] R. Malladi and J. A. Sethian. Level set and fast marching methods in image processing and computer vision. In *ICIP*, volume 1, pages 489–492, 1996.
[27] D. Marcum. *Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection*. CRC Press, Dec. 1998.
[28] M. D. Meyer, R. T. Whitaker, R. M. Kirby, C. Ledergerber, and H. Pfister. Particle-based sampling and meshing of surfaces in multimaterial volumes. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1539–1546, 2008.
[29] P.-O. Persson. Mesh size functions for implicit geometries and PDE-based gradient limiting. *Eng. Comput.*, 22(2):95–109, 2006.

[30] M. S. Shephard and M. K. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *Int. J. Num. Meth. Eng.*, 32(4):709–749, 1991.
[31] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Symp. on Comp. Geom.*, pages 86–95, 1998.
[32] J. R. Shewchuk. Unstructured mesh generation. In U. Naumann and O. Schenk, editors, *Combinatorial Scientific Computing*, chapter 10, pages 257–297. CRC Press, Jan. 2012.
[33] H. Si. TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. http://tetgen.berlios.de/.
[34] M. L. Staten, R. Kerr, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: Progress update. In *15th IMR*, pages 469–486, 2006.
[35] J. Tournois, C. Wormser, P. Alliez, and M. Desbrun. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph.*, 28(3), 2009.
[36] N. P. Weatherill and O. Hassan. Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints. *Int. J. Num. Meth. Eng.*, 37(12):2005–2039, 1994.
[37] S. Yamakawa and K. Shimada. Anisotropic tetrahedral meshing via bubble packing and advancing front. *Int. J. Num. Meth. Eng.*, 57(13):1923–1942, 2003.
[38] M. A. Yerry and M. S. Shephard. Automatic three-dimensional mesh generation by the modified-octree technique. *Int. J. Num. Meth. Eng.*, 20:1965–1990, 1984.
[39] Y. Zhang, T. Hughes, and C. L. Bajaj. Automatic 3D mesh generation for a domain with multiple materials. In *16th IMR*, pages 367–386, 2007.